DP1 2020-2021

Documento de Diseño del Sistema

Proyecto MADAJA

https://github.com/gii-is-DP1/dp1-2020-g1-05

Miembros:

- Echegoyán Delgado, Álvaro
- Pérez Carrillo, Manuel
- Pérez Plata, Juan José
- Pérez Vázquez, Antonio
- Piury Pinzón, Alejandro
- Toro Valle, Daniel

Tutor: Irene Bedilia Estrada Torres

GRUPO G1-05

Versión 1.0

10/01/2021

Historial de versiones

Fecha	Versión	Descripción de los cambios	Sprint
03/01/2021	V1	• Creación del documento	3
09/01/2021	V2	• Añadido diagrama de dominio/diseño • Añadido diagrama de capas • Explicación de la aplicación del patrón MVC • Explicación de las decisiones de diseño	3

Índice

1. Introducción

En este proyecto vamos a desarrollar un sistema de información que se encargue de gestionar correctamente una empresa dedicada al alquiler y venta de vehículos, tanto en establecimientos físicos como de manera online.

Su funcionalidad principal es la de exponer un catálogo de vehículos disponibles, con sus ofertas, por esta empresa para que los clientes puedan realizar y gestionar adecuadamente sus compras y alquileres de manera online, sin necesidad de desplazarse hasta un concesionario físico. También les permite solicitar el envío de un vehículo hasta la localización que deseen además de gestionar los seguros.

No solo ofrece estas facilidades para los clientes, sino que también permite una mejor gestión a nivel interno (para el administrador y para los trabajadores de la empresa) de todos los datos que se manejan.

El objetivo de este sistema es que cumpla todos los requisitos especificados además de que sea fácil de entender y cómodo de usar tanto para clientes como para trabajadores.

2. Diagramas UML

2.1. Diagrama de Dominio/Diseño

Ver Anexo I. Diagrama de Dominio/Diseño.

2.2. Diagrama de Capas

Ver Anexo II. Diagrama de Capas.

3. Patrones de diseño y arquitectónicos aplicados

En esta sección de especificar el conjunto de patrones de diseño y arquitectónicos aplicados durante el proyecto. Para especificar la aplicación de cada patrón puede usar la siguiente plantilla:

3.1. Patrón MVC

Tipo: Arquitectónico

Contexto de Aplicación

El patrón ha sido aplicado a toda la estructura del proyecto, separando los datos de la lógica de negocio, el componente Modelo, aplicado en los paquetes "com.springframework.samples.madaja.model", "com.springframework.samples.madaja.repository" y "com.springframework.samples.madaja.service", la segunda componente Vista, aplicado en las carpetas "src/webapp/WEB-INF/jsp" y "src/webapp/WEB-INF/tags" y el tercer componente, el controlador aplicado en "com.springframework.samples.madaja.web".

Clases o paquetes creados

Se han creado los siguientes paquetes:

- com.springframework.samples.madaja.model:

- Alquiler.java
- Cambio.java
- Cliente.java
- Combustible.java
- Compania.java
- Concesionario.java
- Disponible.java
- Envio.java
- EstadoEnvio.java
- Gestor.java
- Incidencia.java
- Localización.java
- Mecanico.java
- Oferta.java
- Recogida.java
- Reserva.java
- Seguro.java
- SeguroCliente.java
- SeguroVehiculo.java
- Trabajador.java
- User.java
- Vehiculos.java
- Venta.java

- com.springframework.samples.madaja.service:

- AlquilerService.java
- ClienteService.java
- ConcesionarioService.java
- EnvioService.java
- IncidenciaService.java
- MecanicoService.java
- OfertaService.java
- ReservaService.java
- SeguroClienteService.java
- SeguroVehiculoService.java
- UserService.java
- VehiculosService.java
- VentaService.java

- com.springframework.samples.madaja.repository:

- AlquilerRepository.java
- CambioRepository.java
- ClienteRepository.java
- CombustibleRepository.java
- ConcesionarioRepository.java
- ContratoAlquilerRepository.java ?????
- DisponibleRepository.java
- EnvioRepository.java
- EstadoEnvioRepository.java
- IncidenciaRepository.java
- MecanicoRepository.java
- OfertaRepository.java
- ReservaRepository.java
- SeguroClienteRepository.java
- SeguroVehiculoRepository.java
- UserRepository.java
- VehiculosRepository.java
- VentaRepository.java

- src/webapp/WEB-INF/jsp:

- alquiler

        - createAlquilerForm.jsp
        - mostrarMisAlquileres.jsp
- cliente
    - alquilerDetails.jsp
    - mostrarClientes.jsp
    - ventaDetails.jsp
- concesionario
    - concesionarioDetails.jsp
    - mostrarConcesionarios.jsp
- Envio
    - mostrarEnvios.jsp
- incidencia
    - createIncidenciaForm.jsp
    - updateIncidenciaForm.jsp

- oferta
    - createOfertaForm.jsp
    - createOfertaFormAPI.jsp
    - mostrarOfertas.jsp
    - mostrarOfertasAPI.jsp
    - ofertaDetails.jsp
    - updateOfertaForm.jsp

- reservas
    - crearReservaForm.jsp
    - createReservaForm.jsp
    - mostrarMisReservas.jsp
    - mostrarReservas.jsp
    - seleccionarReserva.jsp
- seguro
    - seguroDetails.jsp
- seguroCI
    - createSeguroCIForm.jsp
    - updateSeguroCIForm.jsp
- users
    - createClienteForm.jsp
- vehiculos
    - createOrUpdateVehiculoForm.jsp
    - mostrarVehiculos.jsp
    - vehiculoDetails.jsp
- venta
    - mostrarMisVentas.jsp

Nota: no se han incluido las vistas y elementos que se daban por defecto con la plantilla del proyecto.

Ventajas alcanzadas al aplicar el patrón

Nos permite mantener una gran cantidad de vistas de forma simultánea, manteniendo una cohesión alta, separando la lógica de negocio de la capa de presentación cómodamente.

4. Decisiones de diseño

4.1. Decisión 1: Importación de datos reales para paginaciones

Descripción del problema:

Nos gustaría poder introducir una gran cantidad de datos en el sistema para poder simular casos de uso realistas. El problema es incluir manualmente todos esos datos como parte del script de inicialización de la base de datos.

Alternativas de solución evaluadas:

Alternativa 1.a: Realizar funciones que generen automáticamente (en algunos casos usando una lista de datos externa) una gran cantidad de datos cómo NIF, nombres, listas de vehículos.

Ventajas:

- No requiere introducir manualmente el SQL que genere los datos.
- Más rápido que introducirlos manualmente

Inconvenientes:

- Ralentiza el desarrollo de funcionalidades principales del sistema.
- Tenemos que buscar nosotros listas con datos reales
- Bastante más complejo y tedioso

Alternativa 1.b: Introducir manualmente una cantidad reducida de datos en el SQL pero que sea suficiente para realizar paginaciones.

Ventajas:

- Podemos reutilizar parte de los datos que ya tenemos especificados en (data.sql).
- No afecta al trabajo diario de desarrollo y pruebas de la aplicación
- Es simple, sólo hay que redactar las secuencias SQL

Inconvenientes:

- Tenemos que buscar nosotros listas con datos reales si queremos mantener la coherencia
- La cantidad de datos es mucho menor
- Casos de uso no realistas
- Mucho más lento

Justificación de la solución adoptada

Como consideramos que es fundamental poder aportar casos de uso reales, que reflejen cómo sería el verdadero desempeño del sistema en una situación de este tipo hemos seleccionado la alternativa 1.a.

4.2. Decisión 2: Eliminación de los vehículos

Descripción del problema:

Inicialmente planteamos la eliminación de vehículos del sistema dado que por temas de antigüedad, desgaste o simplemente dar paso a nuevos modelos, el administrador pudiese eliminarlos. El problema que conlleva hacer esto, es que estaríamos eliminando datos que podrían ser de utilidad en un futuro, cómo por ejemplo un precio de referencia para otros modelos del mismo año, qué incidencias eran frecuentes o la demanda del mismo para su posible reintroducción.

Alternativas de solución evaluadas:

Alternativa 2.a: Retirar los vehículos, cambiándoles su disponibilidad a “No disponible”, permitiendo al administrador, dar de alta o de baja un vehículo cuando lo considerase necesario.

Ventajas:

-No se pierde ningún dato que pueda ser útil de cara a un futuro

-Podemos reutilizar parte del trabajo empleado en otros controladores

Inconvenientes:

-Sobrecarga aún más la vista de vehículos, añadiendo una pestaña más y botones, dificultando posibles cambios que sean necesarios

-Es algo complejo, pues los controladores redirigen a la misma página y han de no interactuar a la vez sobre los mismos datos

Alternativa 2.b: Eliminar directamente los vehículos del sistema, cuando el administrador lo considere necesario

Ventajas:

-Muy simple, no interviene con otras funcionalidades

-Complejidad prácticamente nula

Inconvenientes:

-No contempla casos en los que la retirada del vehículo sea solo temporal, cómo por ejemplo un mantenimiento

-Si se necesita en un futuro ese vehículo sería necesario su reintroducción en el sistema

Justificación de la solución adoptada

Consideramos que es necesario no eliminar los vehículos pues pueden ser de utilidad en un futuro y permiten mantener un registro, por lo que tomamos la alternativa 2.a.

4.3. Decisión 3: Seguro cliente

Descripción del problema:

En un principio contemplamos añadir contratos de venta y de alquiler como entidades y en ese contrato de alquiler iría el seguro del cliente. Eliminamos esas dos entidades tras una revisión del proyecto y a la hora de añadir el seguro del cliente nos faltaba una entidad.

Alternativas de solución evaluadas:

Alternativa 3.a: Añadir de nuevo la entidad contrato de alquiler, que iría relacionada con el seguro del cliente.

Ventajas:

-Las historias de usuario varían poco y la planificación sigue igual.

Inconvenientes:

-Añadir una nueva entidad a estas alturas conlleva mucho tiempo y genera conflictos.

Alternativa 3.b: Plantear todo lo relacionado con el seguro del cliente de otra forma. De esta manera el seguro del cliente va asociado directamente al vehículo y lo crea, edita o elimina el gestor. El cliente puede alquilar un vehículo con un seguro de cliente solo si se ha asociado un seguro a dicho vehículo.

Ventajas:

-La modificación de las historias de usuario es relativamente pequeña.

Inconvenientes:

-Todo el proceso relacionado con el seguro del cliente es un poco más complejo.

Justificación de la solución adoptada

Debido al tiempo y los conflictos que conllevaría añadir una nueva entidad hemos optado por la alternativa 3.b.

4.4. Decisión 4: Introducción sistema de auditoría

Descripción del problema:

Pensamos que la auditoría sería una funcionalidad muy interesante para implementar puesto que permitiría al administrador llevar un control de los cambios que se produzcas en diferentes apartados de la web, cómo las ofertas, modificaciones en los datos de los vehículos, etc. Problema, es una funcionalidad adicional no requerida para la evaluación del proyecto y puede ralentizar el desarrollo de otras funcionalidades sí requeridas.

Alternativas de solución evaluadas:

Alternativa 4.a: Incluir el sistema de auditoría, dividiendo el trabajo a realizar en el sistema, mientras un grupo desarrolla las funcionalidades principales necesarias, 1-2 personas se dedican al desarrollo de dicho sistema.

Ventajas:

-Es una funcionalidad extra que consideramos importante en un sistema real y creemos que su introducción puede ser valorada positivamente

-Añade versatilidad al proyecto

-No es demasiado complejo

Inconvenientes:

-Ralentiza el desarrollo de funcionalidades principales

-Puede generar algún conflicto con funcionalidades ya implementadas

Alternativa 4.b: No incluirlo directamente e invertir ese tiempo en otras funcionalidades principales u otras características cómo la estética de la página web y las vistas.

Justificación de la solución adoptada

Aún no hemos decidido completamente, pero en un principio hemos tomado la alternativa 4.a.

4.5. Decisión 5: Eliminación de ofertas

Descripción del problema:

A la hora de programar la parte relacionada con las ofertas nos surgió la duda de si era mejor eliminar las ofertas o simplemente ocultarlas o archivarlas.

Alternativa 5.a: Eliminar las ofertas.

Ventajas:

-Si hay alguna oferta errónea u obsoleta podría ser eliminada de por vida

Inconvenientes:

-No existiría un historial de ofertas ni la posibilidad de reutilizar una misma oferta en diferentes momentos.

Alternativa 5.b: Ocultar o archivar las ofertas.

Ventajas:

-Existiría un historial de ofertas que nos permitiría consultar ofertas pasadas y podría utilizarse una misma oferta en varias ocasiones, puesto que las ofertas archivadas u ocultas se podrían mostrar de nuevo.

Inconvenientes:

-Si alguna oferta es errónea o queda obsoleta no podría borrarse.

Justificación de la solución adoptada

Hemos optado por la alternativa 5.b porque pensamos que desde el punto de vista de la funcionalidad es útil tener un historial de todas las ofertas, por lo que no es necesario borrarlas, así como la posibilidad de reutilizar una oferta varias veces.

4.6. Decisión 6: Asignación de vehículos a una oferta

Descripción del problema:

Con relación al apartado de ofertas, se nos planteó la duda de si una oferta se asociaría a un solo vehículo, o, por el contrario, esta podía estar asociada a más de uno.

Alternativa 6.a: Una oferta está asociada a un solo vehículo.

Ventajas:

-Independencia de las ofertas. Cada oferta se corresponde con un solo vehículo

-Se mantienen las relaciones planteadas en el modelado

Inconvenientes:

-Pueden haber varias ofertas iguales asociadas cada una a un vehículo diferente

Alternativa 6.b: Una oferta puede estar asociada a varios vehículos.

Ventajas:

-Disminuye la cantidad de ofertas, ya que una oferta puede referirse a varios vehículos. No es necesario crear siempre una nueva oferta

Inconvenientes:

-Puede generar confusión, ya que cuando miramos una oferta temenos que fijarnos en todos los vehículos a los que está asignada

-Es necesario cambiar algunas relaciones para poder llevar a cabo la alternativa

Justificación de la solución adoptada

Pensamos que es más útil poder asignar una misma oferta a varios vehículos y evitar de esta manera las ofertas repetidas. Además los cambios en las relaciones no suponían grandes conflictos ni mucho tiempo, por lo que hemos optado por la opción 6.b.

4.7. Decisión 7: Eliminación de entidades de contratos

Descripción del problema:

Cuando teníamos todo el proyecto modelado nos dimos cuenta de la existencia de dos entidades que carecían de atributos y representaban los contratos de alquiler y venta. Tras una revisión del proyecto llegamos a la conclusión de que los contratos eran archivos más que entidades y necesitábamos encontrar una forma de solucionar este problema.

Alternativas de solución evaluadas:

Alternativa 7.a: Eliminar las dos entidades y considerar contratos como archivos.

Ventajas:

-Eliminamos entidades sin atributos que carecen de sentido

Inconvenientes:

-Llevar a cabo los cambios en el resto del modelado

Alternativa 7.b: Buscar atributos para ambas entidades.

Ventajas:

-No hay que modificar nada del modelado

Inconvenientes:

-Tener entidades poco útiles con atributos innecesarios

Justificación de la solución adoptada

Hemos optado por la alternativa 7.a porque de esta manera eliminábamos entidades que resultaban poco útiles en el proyecto.

4.8. Decisión 8: Cambio de la relación de oferta y otras entidades

A medida que íbamos implementando varias historias de usuario nos iban surgiendo problemas y para solucionarlos y llevar a cabo los cambios tuvimos que cambiar entidades ya implementadas.

Alternativa 8.a: Cambiar las relaciones de algunas entidades (oferta, seguroCliente, cliente, compañía, vehiculo y alquiler).

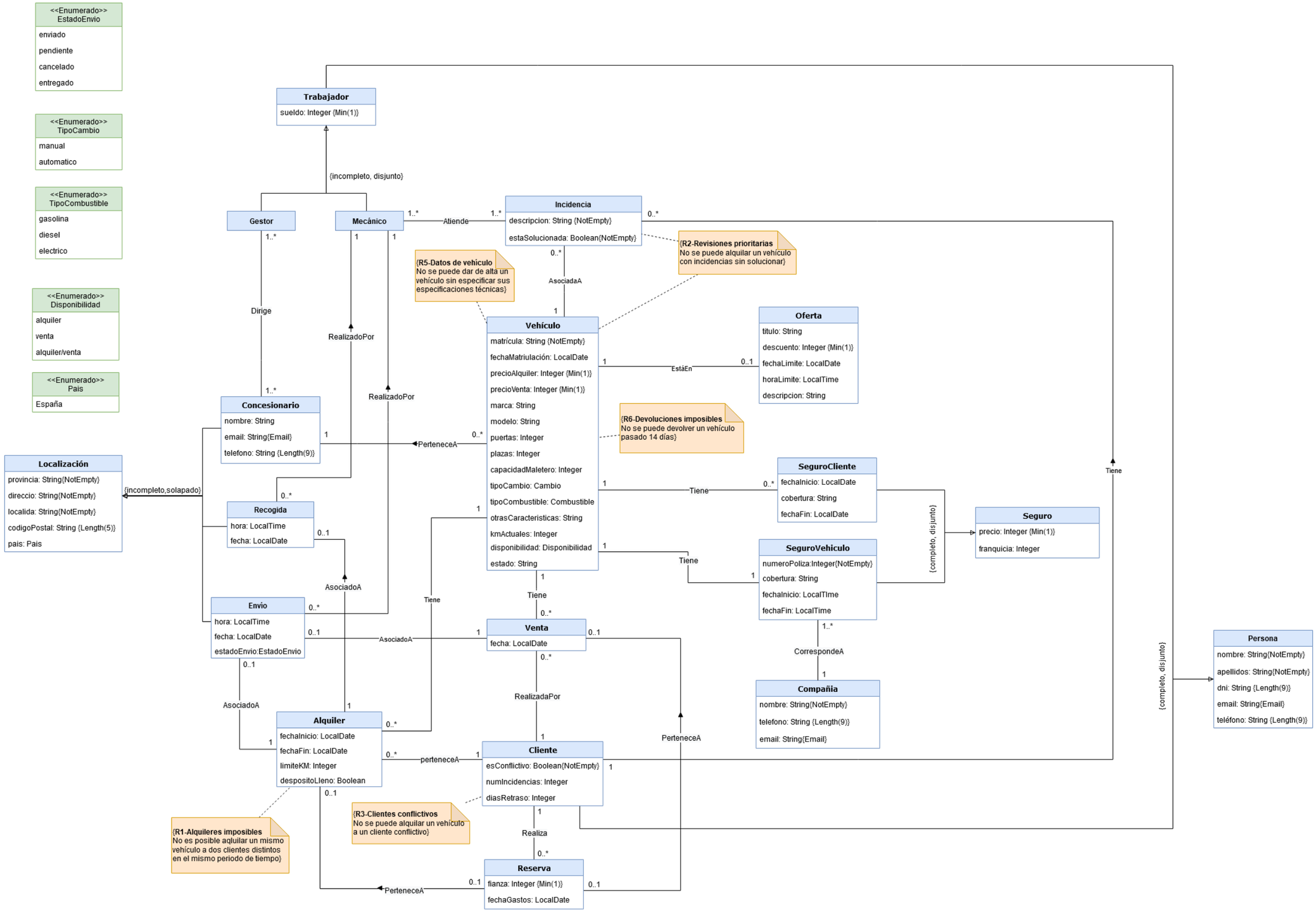
Ventajas:

-Solución rápida para poder implementar los cambios sugeridos.

-No conlleva mucho tiempo.

-No genera grandes conflictos

Anexo I. Diagrama de Diseño/Dominio



Anexo II. Diagrama de Capas